QUARKUS

Java
Rest Services

JEE Microservices

@ CGS IT – 2023

Version 1.0.5

Inhalt

- Rest Standards
- Rest Methoden
- JaxRs Library
- Java Beispiele
- Erweiterte Dokumentation
- Dokumentations Links

REST : Definition [Wikipedia]

A **REST API** is an application programming interface (**API**) that uses a representational state transfer (**REST)** architectural style. The REST architectural style uses HTTP to request access and use data. This allows for interaction with RESTful web services.

- Representational State Transfer
- A style of software architecture for distributed hypermedia systems such as the World Wide Web.
- was introduced and defined in 2000 by Roy Fielding in his doctoral dissertation.
- Conforming to the REST constraints is referred to as being ‘RESTful’.

Rest is

- An architectural style, not technology
- Client/server + Request/response approach.
- Everything is a RESOURCE.
- CRUD (Create / Read / Update / Delete)
- Stateless by nature (excellent for distributed systems)
- Cacheable (naturally supported)
- A great way to web-service

Rest Methods

Method	Description
GET	Retrieve information about the REST API resource
POST	Create a REST API resource
PUT	Update a REST API resource
DELETE	Delete a REST API resource or related component

Rest API URL Design

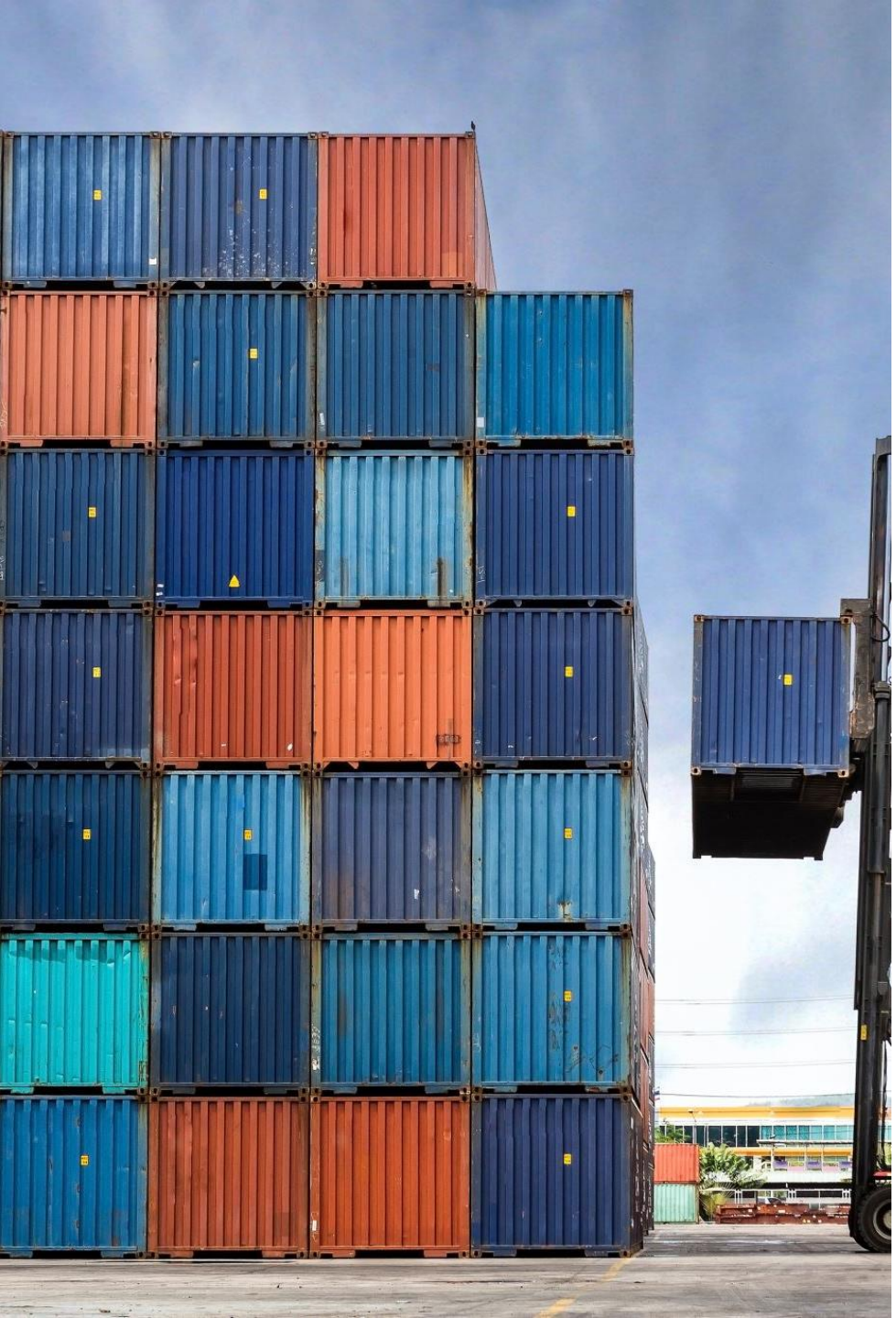
Method		
GET	/tickets	Retrieves a list of tickets
GET	/tickets/12	Retrieves a specific ticket
POST	/tickets	Creates a new ticket
PUT	/tickets/12	Updates ticket #12
DELETE	/tickets/12	Deletes ticket #12

Jax-RS
Rest Services

Jax-RS Definition - Wikipedia

- Bei den **Jakarta RESTful Web Services** (JAX-RS; handelt es sich um die Spezifikation einer Programmierschnittstelle (API) der Programmiersprache Java,
- die die Verwendung des Software-Architekturstils Representational State Transfer (REST) im Rahmen von Webservices ermöglicht und vereinheitlicht.
- Die in der Spezifikation beschriebenen Funktionalitäten wurden vom Java Community Process erarbeitet
- und im Java Specification Request 311[1] verabschiedet.
- Wie auch andere Programmierschnittstellen der Jakarta EE (JEE) benutzt JAX-RS Annotationen, um die Entwicklung und das Deployment von Webservice-Clients und Service-Endpunkten zu vereinfachen.

https://de.wikipedia.org/wiki/Jakarta_RESTful_Web_Services



JAX-RS (JSR-311) : Goals

- POJO-based
- HTTP-centric
- Format independent
- Container independent
- Availability as standalone and enterprise platforms

Jaxrs - Resteasy

- RESTEasy is a JBoss / Red Hat project that provides various frameworks to help you build RESTful Web Services and RESTful Java applications.
- It is an implementation of the Jakarta RESTful Web Services, an Eclipse Foundation specification that provides a Java API for RESTful Web Services over the HTTP protocol.
- Moreover, RESTEasy also implements the MicroProfile REST Client specification API.
- RESTEasy can run in any Servlet container, WildFly Application Server and Quarkus is also available to make the user experience nicer in those environments.

https://resteasy.dev/

Jaxrs & Resteasy Documentation

- https://jcp.org/en/jsr/detail?id=311
- https://docs.jboss.org/resteasy/docs/6.2.2.Final/userguide/html_single/index.html
- Cheat Sheet für Resteasy Annotations

https://www.mastertheboss.com/jboss-frameworks/resteasy/jax-rs-cheatsheet/

Using @Path and @GET, @POST, etc.

```
@Path("/library")
public class Library {

   @GET
   @Path("/books")
   public String getBooks() {...}

   @GET
   @Path("/book/{isbn}")
   public String getBook(@PathParam("isbn") String id) {
      // search my database and get a string representation and return it
   }

   @PUT
   @Path("/book/{isbn}")
   public void addBook(@PathParam("isbn") String id, @QueryParam("name") String name) {...}

   @DELETE
   @Path("/book/{id}")
   public void removeBook(@PathParam("id") String id {...}

}
```

RESTEasy servlet is configured and reachable at a root path of http://myhost.com/services.

The requests handled by class, Library, are:

- GET http://myhost.com/services/library/books
- GET http://myhost.com/services/library/book/333
- PUT http://myhost.com/services/library/book/333
- DELETE http://myhost.com/services/library/book/333

https://docs.jboss.org/resteasy/docs/6.2.2.Final/userguide/html_single/index.html#Using_Path

@PathParam

```
@Path("/parameter")
public class ParameterResource { }
```

```
http://localhost:8080/json/inputParameter/{inputString}
no usages
@GET
@Path("/inputParameter/{inputString}")
@Produces(MediaType.TEXT_PLAIN)
public String inputParameter(@PathParam("inputString") String inputString){
    log.infov( format: "log: {0}", inputString);
    return inputString;
}
```

CGS ITSolutions

Lösung 1

```
@Path("/parameter")
public class ParameterResource {

   @Inject
   Logger log;

   @GET
   @Path("/")
   @Produces(MediaType.TEXT_PLAIN)
   public String basePath(){
      log.infov("basePath");
      return "basePath";
   }

   @GET
   @Path("/inputParameter/{inputString}")
   @Produces(MediaType.TEXT_PLAIN)
   public String inputParameter(
         @PathParam("inputString") String inputString){
      log.infov("log: {0}", inputString);

      StringBuilder sbStr = new StringBuilder();
      sbStr.append(inputString).reverse();
      return sbStr.toString();
   }
```

CGS ITSolutions

@QueryParam

```
@Path("/parameter")
public class ParameterResource { }
```

```
http://localhost:8080/json/queryParameter
// http://localhost:8080/json/queryParameter?qp=inputText&qp2=text2
no usages
@GET
@Path("/queryParameter")
@Produces(MediaType.TEXT_PLAIN)
public String queryParameter(
        @QueryParam("qp") String qp,
        @QueryParam("qp2") String qP2
        ){
    log.infov( format: "log QueryParam: {0}", qp);
    return qp + " und " + qP2;
```

Query Parameter – 2

Query Parameter als Long :

```
public String queryParameter(
        @QueryParam("qp") String qp,
        @QueryParam("qp2") Long qP2
```

1. Wenn ein Parameter nicht übergeben wird ist is „null“
2. Wenn ein Input Parameter nicht konvertiert werden kann, wird kein Fehler geworfen, sondern die Methode nicht gefunden.
    1. Test: String input für qP2 sollte die Methode nicht finden

Default Werte für Query Parameter

```
@GET
@Path("/queryParameter")
@Produces(MediaType.TEXT_PLAIN)
public String queryParameter(
        @QueryParam("qp") String qp,
        @DefaultValue(„5") @QueryParam("qp2") Long qP2
        ){
    log.infov("log QueryParam: {0}", qp);
    return "query params ["+ qp + "] und [" + qP2 + "]";
}
```

Spezial Informationen

```
@GET
@Path("header")
public String checkBrowser(@HeaderParam("User-Agent")
String whichBrowser) {
   return "Browser is "+whichBrowser;
}

// Reading REST Parameters Programmatically
// z.b. http://localhost:8080/json/context?username=chris
@GET
@Path("context")
public Response login(@Context UriInfo info) {
   String id = info.getQueryParameters().getFirst("username");
   return Response
         .status(200)
         .entity("login called with id: " + id)
         .build();
}
```

Try and Test APIs

- OpenAPI / Swagger dependency erlaubt das einfache testen direkt via Browser unter:

http://localhost:8080/q/swagger-ui/

```
<!-- add openapi swagger support -->
<dependency>
   <groupId>io.quarkus</groupId>
   <artifactId>quarkus-smallrye-openapi</artifactId>
</dependency>
```

Form Param & @Form

- @Form is a RESTEasy specific annotation that allows the re-use of any @*Param annotation within an injected class.
- RESTEasy will instantiate the class and inject values into any annotated @*Param or @Context property.
- This is useful when there are a lot of parameters on a method and it is wanted to condense them into a value object.

Siehe:

- https://docs.jboss.org/resteasy/docs/6.2.2.Final/userguide/html_single/index.html#Using_Path

Jakarta Content Negotiation

- The HTTP protocol has built in content negotiation headers that allow the client and server to specify what content they are transferring and what content they would prefer to get. The server declares content preferences via the @Produces and @Consumes headers.

@Produces and @Consumes

The @Produces is used to map a client request and match it up to the client's Accept header. The Accept HTTP header is sent by the client and defines the media types the client prefers to receive from the server.

```
@Produces("text/*")
@Path("/library")
public class Library {

@GET
@Produces("application/json")
public String getJSON() {...}

@GET
public String get() {...}
```

So, if the client sends:

```
GET /library
Accept: application/json
```

CGS ITSolutions

JSON Support via Jackson Project

```
<dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-resteasy</artifactId>
</dependency>
<!-- add json output support via jackson library -->
<dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-resteasy-jackson</artifactId>
</dependency>
```

Aufgabe 2:

- RestFull Compatible GET
- READ /{id}
  - GET /testdto/11 -> liefert das objekt mit der ID 11
- List All wenn keine ID angeben
  - GET /testdto → liefert alle Objekte
  - public List<TestDTO> listAllObects(){

```
@GET
@Path("/")
@Produces({MediaType.APPLICATION_JSON,
MediaType.APPLICATION_XML})
public List<TestDTO> listAllObects(){
   log.infov("objectOutput {0}", "");

   TestDTO dto = new TestDTO();
   dto.setName("name");
   dto.setVorname("voranme");

   TestDTO dto2 = new TestDTO();
   dto.setName("name2");
   dto.setVorname("voranme2");

   List result = new ArrayList();
   result.add(dto);
   result.add(dto2);

   return result;
}
```

CGS ITSolutions

Jaxrs Resteasy Advanced Features

- CORS Support
- Multipart Provider
- Security
- Zip Support
- Exception Handlin

See Resteay User-Guide:

https://docs.jboss.org/resteasy/docs/6.2.2.Final/userguide/html_single/index.html#d5e2342

Json – Object Output

@Produces gibt an welches Output Format erzeugt wird.

localhost:8080/json/objectOutput

```
@Path("/json")
public class JsonApiResource {

   @Inject
   Logger log;

   @GET
   @Path("/objectOutput")
   @Produces(MediaType.APPLICATION_JSON)
   public TestDTO objectOutput(){
      log.infov("objectOutput {0}", "");

      TestDTO dto = new TestDTO();
      dto.setName("name");
      dto.setVorname("voranme");

      return dto;
   }
```

CGS ITSolutions

Json – Object Input

- @Consumes(MediaTyp e.*APPLICATION_JSON*)

Gibt an welches Object als Input übernommen werden soll

```
@PUT
@Path("createTestMessage")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
public TestDTO updateTestMessage(TestDTO input) {
   log.infov("got object : {0}", input.toString());

   return input;
}
```

CGS ITSolutions

Danke für Ihre Aufmerksamkeit

CGS ITSolutions